**CODE 1:**

import numpy as np

c1 = [1,1,1,1]

c2 = [1,-1,1,-1]

c3 = [1,1,-1,-1]

c4 = [1,-1,-1,1]

rc = []

print("Enter the data bits")

d1 = int(input("Enter D1: "))

d2 = int(input("Enter D2: "))

d3 = int(input("Enter D3: "))

d4 = int(input("Enter D4: "))

r1 = np.multiply(c1, d1)

r2 = np.multiply(c2, d2)

r3 = np.multiply(c3, d3)

r4 = np.multiply(c4, d4)

resultant\_channel = r1 + r2 + r3 + r4

print("Resultant Channel ", resultant\_channel)

Channel = int(input("Enter the station to listen for C1=1, C2=2, C3=3, C4=4"))

if Channel==1:

rc = c1

elif Channel==2:

rc = c2

elif Channel==3:

rc = c3

elif Channel==4:

rc = c4

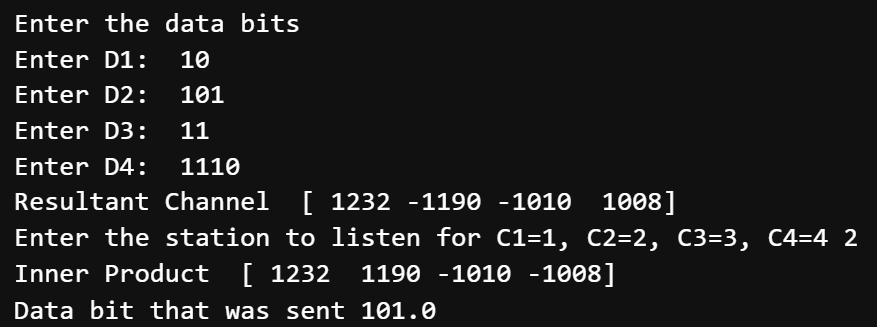
inner\_product = np.multiply(resultant\_channel, rc)

print("Inner Product ", inner\_product)

res1 = sum(inner\_product)

data = res1/len(inner\_product)

print("Data bit that was sent",data)

**OUTPUT:  
**

**CODE 2 (using Walsh Code):**

import numpy as np

# FUNCTION DEFINITIONS

def generate\_walsh\_code(order):

if order== 1:

return np.array([[1, 1], [1, -1]])

else:

previous\_matrix = generate\_walsh\_code(order-1)

upper = np.concatenate((previous\_matrix, previous\_matrix), axis=1)

lower = np.concatenate((previous\_matrix, -previous\_matrix), axis=1)

return np.concatenate((upper, lower), axis=0)

def encode(data\_bits, walsh\_matrix):

encoded\_result = np.zeros\_like(walsh\_matrix[0]) #Initializes encoded\_result as an array of zeros with the same shape as the first row of the Walsh matrix.

for i, bit in enumerate(data\_bits):

encoded\_result += bit \* walsh\_matrix[i]

return encoded\_result

def decode(encoded\_data, walsh\_matrix, channel):

decoded\_result = np.dot(encoded\_data, walsh\_matrix[channel]) / len(walsh\_matrix[channel]) #Decodes the data for the specified channel by taking the dot product of the encoded data with the Walsh code of the channel and dividing by the length of the Walsh code.

return decoded\_result

# ACCEPTING USER INPUT

bit\_count = int(input("\nEnter the number of data bits :"))

data\_bits = []

for i in range(bit\_count):

bit = int(input(f"\nEnter the bit number {i + 1}: "))

data\_bits.append(bit)

# GENERATING WALSH CODE

x=1

print("x before: ",x)

while (2\*\*x<bit\_count):

x=x+1 #logx/log2

print("x after: ",x)

walsh\_matrix = generate\_walsh\_code(x)

np.set\_printoptions(threshold=np.inf) # Set printing options

print("\nEntered data is :")

print(data\_bits)

print("\nGenerated Walsh Matrix:")

print(walsh\_matrix)

# ENCODING DATA

encoded\_data = encode(data\_bits, walsh\_matrix)

print(f"\nEncoded data is: {encoded\_data}")

# DECODING DATA

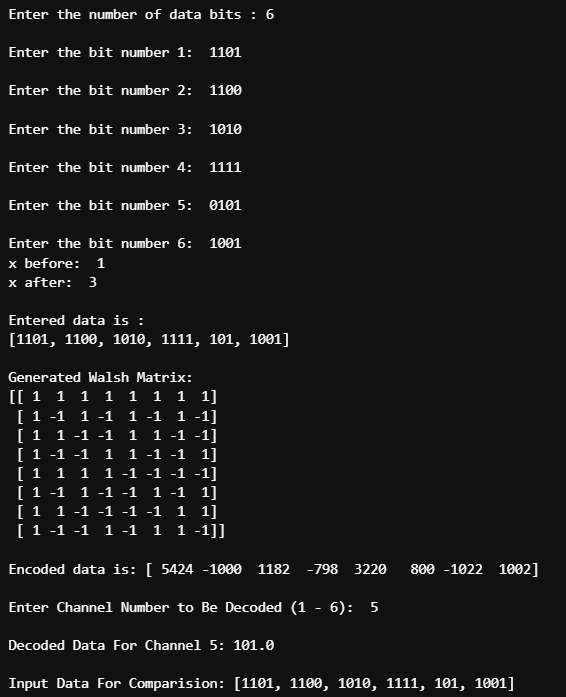
dChannel = int(input(f"\nEnter Channel Number to Be Decoded (1 - {bit\_count}): "))

decoded\_data = decode(encoded\_data, walsh\_matrix, dChannel-1)

print(f"\nDecoded Data For Channel {dChannel}: {decoded\_data}")

print(f"\nInput Data For Comparision: {data\_bits}")

**OUTPUT:**

****